Universitat de Girona

Pràctica:
"Introducció a la programació en JavaSoccer"

Disseny de Sistemes de Supervisió (5è EInf)

Dept. Electrònica Informàtica i Automàtica

Pràctica: “Introducció a la programació en JavaSoccer”

Objectius de la pràctica:

- Conèixer l’entorn de JavaSoccer.
- Aprendre els mètodes bàsics per a programar en aquest entorn.

1 Introducció

JavaSoccer és un programa escrit en Java i desenvolupat per Tucker Balch a la Universitat de Carnegie Mellon, que simula la lliga de futbol de robots petits de la RoboCup, tant en la dinàmica (comportament del cos físic) com en les mides dels robots.

En aquesta competició intervenen 5 robots per equip que competeixen en un camp amb moqueta, empenyent i xutant una pilota de golf dins la porteria de l’equip contrari.

La **Figura 1** mostra la pantalla principal d’aquest programa, amb la configuració de sortida dels robots en el camp al començament del partit.

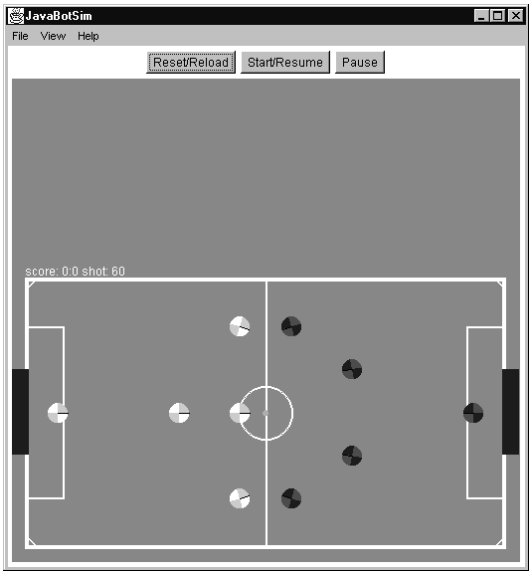


Figura 1: Pantalla principal i posicions dels jugadors al començament del partit.

Aquest programa permet manipular les posicions inicial, la grandària i els colors dels jugadors. A més a més, permet visualitzar el número de jugador, les seves traces i l’acció que està fent en cada moment.

2 Característiques de JavaSoccer.

Les dimensions del camp de futbol són de 152.2 cm d'amplada per 274 cm de llargada. Cada porteria té una mida de 50 cm. En el sistema de coordenades utilitzat en la simulació, el centre del camp és el (0,0) amb +x cap a la dreta (est) i +y cap a dalt (nord). L'equip que comença a la part dreta del camp s'anomena equip "east" i el de l'esquerra, equip "west".

Els robots són rodones de 12 cm de diàmetre i poden moure's a una velocitat lineal de 0.3 m/s i girar 360 graus per segon. Cadascú té un número d'identificació i un equip pot tenir com a màxim, cinc robots. Ells poden lliscar seguint la paret sempre i quan no hi estiguin encarats.
La pilota té un diàmetre de 4 cm i pot arribar a una velocitat màxima de 0.5 m/s quan es xuta. Desaccelera a 0.1 m/s^2. Les col·lisions amb la paret són perfectament elàstiques. Actualment no hi ha cap restricció en el temps de durada del partit, però la RoboCup estableix un temps de 10 minuts.

Cada porteria està envoltada per una zona de defensa, que es troba a 22.5 cm de la línia de gol i que té una amplada de 100 cm. Només un robot defensor pot entrar en aquesta àrea.

Malgrat que aquest programa té assignades unes posicions inicials dels jugadors al començament del partit, la RoboCup no n'especifica cap, però sí, que l'equip que xuta primer pot ésser a menys de 15 cm de la pilota, mentre que l'equip contrari ha de ser fora de la rodona central.

JavaSoccer no fa acomplir cap tipus de faltes, però existeix un rellotge que comença a funcionar quan es fa un gol i que conta enrera des de 60 segons. Si cap equip anota en aquest temps, la pilota torna automàticament al centre del camp. La pilota també torna al centre si queda encallada a la paret o entre jugadors durant un cert temps.

3 Programació en JavaSoccer.

Com s'ha dit a la introducció, JavaSoccer està programat en Java, per tant, la majoria de coses es pot fer en aquest llenguatge de programació. Però, cal conèixer unes certes funcions que permeten seguir l'evolució del partit.

La classe principal es l'abstract_robot. En aquesta estan contingudes totes les funcions necessàries per comandar el robot i per saber la situació de cadascun d'ells. La majoria dels paràmetres que s'utilitzen en aquesta classe principal, pertanyen a una altra, anomenada Vec2.

3.1 Classe Vec2.

La classe Vec2, s'utilitza per a manipular vectors de dos dimensions, i sempre estan disponibles tant les coordenades cartesianes **(x , y)** com les polars **(t, r)** , on t és l'angle en radians i r el radi. Les coordenades x, y i el radi r estan en metres. Una coordenada

+x (positiva) és a la dreta del centre del camp mentre que +y és cap a dalt. Per una altra banda, t és 0 en la direcció de +x i PI en la de –x, avançant en sentit anti-horari (tant per a un jugador de l'est com de l'oest). Aquest paràmetres es poden canviar utilitzen els mètodes setx, sety, sett i setr respectivament.

Aquesta classe té 3 tipus de constructors diferents.

- **Vec2 ():**
  Les variables tenen per defecte els valors següents:
  x = 1; y = 0;
  t = 0; r =1;

- **Vec2 (double x0, double y0)**
  Les variables tenen els valors següents:
  x = x0; y = y0;
  t = Math.atan2(y,x); r = Math.sqrt(x*x + y*y);

- **Vec2 (Vec2 v)**
  On les variables valen:
  x = v.x; y = v.y;
  t = v.t; r = v.r;

Els mètodes associats a Vec2 són:

- **setx (double noux):** canvia el valor de la x i actualitza el radi i l'angle.
- **sety (double nouy):** modifica el valor de la y i recalcula el radi i l'angle.
- **sett (double nout):** canvia l'angle i reemplaça els valors de la x i de la y.
- **setr (double nour):** modifica el valor de r i actualitza els valors de la x i de la y.
- **rotate (double t1):** suma al valor de l'angle el de rotació t1 i recalcula x i y. Fa el mateix que sett (t + t1).
- **sub (Vec2 v2):** resta el vector v2 a ell mateix (this = this – v2).
- **normalize (double r1):** fa el mateix que setr.
- **add(Vec2 v2):** suma el vector v2 a ell mateix (this = this + v2).
- **octant ():** retorna un valor entre 0 i 7 i serveix per saber en quin octant (vuitena part d'una circumferència), es troba apuntant el vector. El 0 indica 0 radians i seguint (± PI/8) en sentit anti-horari.
- **quadrant ():** retorna un valor entre 0 i 3 i serveix per saber en quin quadrant de la circumferència apunta el vector. El 0 indica 0 radians i seguint (± PI/4) en sentit anti-horari.
- **toString():** genera una cadena que conté els valors del vector: "(x, y) (r, t)"

3.2 Classe abstract_robot.

Aquesta classe emula diferents classes de robots, i la que s'ampliarà per a desenvolupar l'estratègia de joc és la ControlSystemSS. Al manual d'usuari de JavaSoccer es poden trobar tots els mètodes que aquesta classe té, no obstant a continuació s'enumeraren els principals.

- **getTime():** retorna el temps que ha passat des que el robot ha estat inicialitzat.
- **getPosition (long current_time):** obté la posició del robot en coordenades globals (referides al centre del camp).

- **getBall (long current_time):** obté una variable Vec2 que apunta a la pilota.
- **getOutGoal (long current_time):** retorna un Vec2 que apunta a la porteria de l'equip.
- **getOpponentsGoal (long current_time):** retorna un Vec2 que apunta a la porteria contraria.
- **getTeammates (long current_time):** retorna un array de Vec2 que apunta egocèntricament des del centre del robot als companys sensats.
- **getOpponents (long current_time):** retorna un array de Vec2 que apunta egocèntricament des del centre del robot als contraris sensats.
- **getPlayerNumber (long current_time):** obté un sencer que representa la identificació del robot en l'equip.
- **getSteerHeading (long current_time):** retorna la orientació actual del robot en radians.
- **setSteerHeading (long current_time, double angle):** permet fixar l'orientació desitjada del robot.
- **setSpeed (long current_time, float speed):** posa la velocitat de translació desitjada del robot.
- **setDisplayString (String):** permet posar el comentari que es desitja aparegui amb el robot.
- **canKick (long current_time):** permet saber si la pilota està en una posició correcta per a ser xutada.
- **kick (long current_time):** per a xutar la pilota si aquesta és a la posició correcta.

A més d'aquests mètodes, existeixen uns altres que permeten la comunicació entre els robots, agafar i deixar objectes, si un objecte és un obstacle, etc..

Totes aquests mètodes es poden trobar al manual d'usuari, i pertanyen a la llibreria **SocSmallSim** de l'**abstractrobot**. Aquí també, es troba el mètode **TakeStep()**, que és el que realitza la simulació del robot.

3.3 Creació del fitxer amb l'estratègia de joc.

Per poder treballar amb JavaSoccer, es necessari crear un fitxer .java amb el mateix nom que la classe en la qual s'escriurà l'estratègia de joc de cada jugador.

1) el fitxer ha de començar important algunes llibreries com són:

- java.io.*
- java.lang.Math
- EDU.gatech.cc.is.util.Vec2
- EDU.gatech.cc.is.abstractrobot.*

Aquestes llibreries s'inclouen utilitzant la paraula ***import***.

2) Definició de la nova classe derivada de la ControlSystemSS:

```
public class nom extends ControlSystemSS
{
	/* Aquí es poden fer totes les inicialitzacions que calguin (només una vegada, quan es crea la classe) per a utilitzar les diferents eines de l'IA .*/
	public void Configure()
	{
		/* Aquest pot ésser rescrit si es vol configurar un sistema de control propi.*/
	}
	public int TakeStep()
	{
		/* Aquí s'ha d'escriure tot el codi de l'estratègia de joc */
	}
}
```

4 Exemple de Classe (GoToBall.java).

```
package JavaSoccer.teams;

import EDU.gatech.cc.is.util.Vec2;
import EDU.gatech.cc.is.abstractrobot.*;

public class GoToBall extends ControlSystemSS
{
	public void Configure()
	{
	}

	public int TakeStep()
	{
		Vec2   result,ball;
		long   curr_time = abstract_robot.getTime();

		// get vector to the ball
		ball = abstract_robot.getBall(curr_time);

		// set heading towards it
		abstract_robot.setSteerHeading(curr_time, ball.t);

		// set speed at maximum
		abstract_robot.setSpeed(curr_time, 1.0);

		// kick it if we can
		if (abstract_robot.canKick(curr_time))
			abstract_robot.kick(curr_time);

		// tell the parent we're OK
		return(CSSTAT_OK);
	}
}
```

5 Execució del Programa.

Per executar aquest programa cal cridar al fitxer **demo.bat** que es troba al directori /javasoccer/javasoccer. Aquest en fa servir un altre anomenat **robocup.dsc** que conté els paràmetres inicials del simulador, com són les posicions inicials dels jugadors, els colors dels equips, la posició de la pilota (al centre del camp), les mides del camp, les parets, etc., i un de molt important que és el nom de la classe Java que conté l'estratègia de l'equip.

Per més informació consulteu la pàgina web:

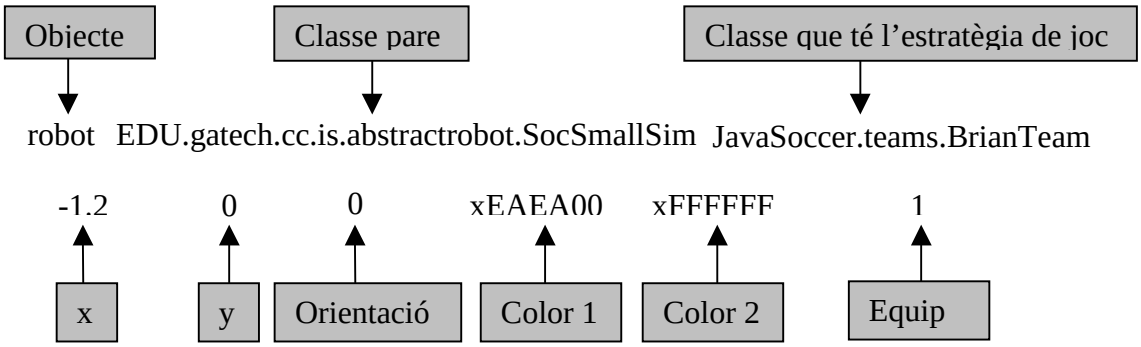
http://eia.udg.es/~iitap/cursjava/javasoccer.html

5.1 Fitxer Robocup.dsc.

Aquest fitxer conté tota la informació necessària per a manipular l'entorn gràfic, és a dir, la pantalla principal. Al final d'aquest es troba la part que interessa, on es pot modificar el comportament dels robots. A continuació es mostra aquesta part del fitxer.

```
//======
// ROBOTS
//======
//======WEST TEAM======
robot EDU.gatech.cc.is.abstractrobot.SocSmallSim JavaSoccer.teams.BrianTeam
//------------your control system name goes here ^^^^^^^^
        -1.2  0    0 xEAEA00 xFFFFFF 1
robot EDU.gatech.cc.is.abstractrobot.SocSmallSim JavaSoccer.teams.BrianTeam
//------------your control system name goes here ^^^^^^^^
        -.5   0    0 xEAEA00 xFFFFFF 1
robot EDU.gatech.cc.is.abstractrobot.SocSmallSim JavaSoccer.teams.BrianTeam
//------------your control system name goes here ^^^^^^^^
        -.15  .5   0 xEAEA00 xFFFFFF 1
robot EDU.gatech.cc.is.abstractrobot.SocSmallSim JavaSoccer.teams.BrianTeam
//------------your control system name goes here ^^^^^^^^
        -.15  0    0 xEAEA00 xFFFFFF 1
robot EDU.gatech.cc.is.abstractrobot.SocSmallSim JavaSoccer.teams.Kechze
//------------your control system name goes here ^^^^^^^^
        -.15  -.5  0 xEAEA00 xFFFFFF 1

//======EAST TEAM======
robot EDU.gatech.cc.is.abstractrobot.SocSmallSim JavaSoccer.teams.SchemaDemo
         1.2 0    0 xFF0000 x0000FF 2
robot EDU.gatech.cc.is.abstractrobot.SocSmallSim JavaSoccer.teams.SchemaDemo
         .5   .5    0 xFF0000 x0000FF 2
robot EDU.gatech.cc.is.abstractrobot.SocSmallSim JavaSoccer.teams.SchemaDemo
         .15  .5   0 xFF0000 x0000FF 2
robot EDU.gatech.cc.is.abstractrobot.SocSmallSim JavaSoccer.teams.SchemaDemo
         .5  0    0 xFF0000 x0000FF 2
robot EDU.gatech.cc.is.abstractrobot.SocSmallSim JavaSoccer.teams.SchemaDemo
         .15  -.5  0 xFF0000 x0000FF 2
```



Les coordenades negatives es deuen a que el centre de coordenades és al mig del camp. En aquest fitxer s'ha de canviar la classe que té l'estratègia de futbol, i si es vol els colors d'equip. També es poden ubicar els jugadors en les posicions desitjades.